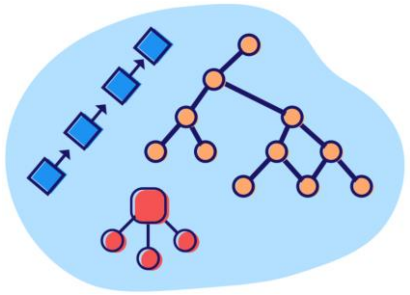
الجامعة العربية الخاصة للعلوم والتكنولوجيا
كلية الهندسة المعلوماتية
قسم تقانة المعلومات

Algorithms & Data Structures (2)

Lecture-8: AVL Trees

(أشجار AVL)



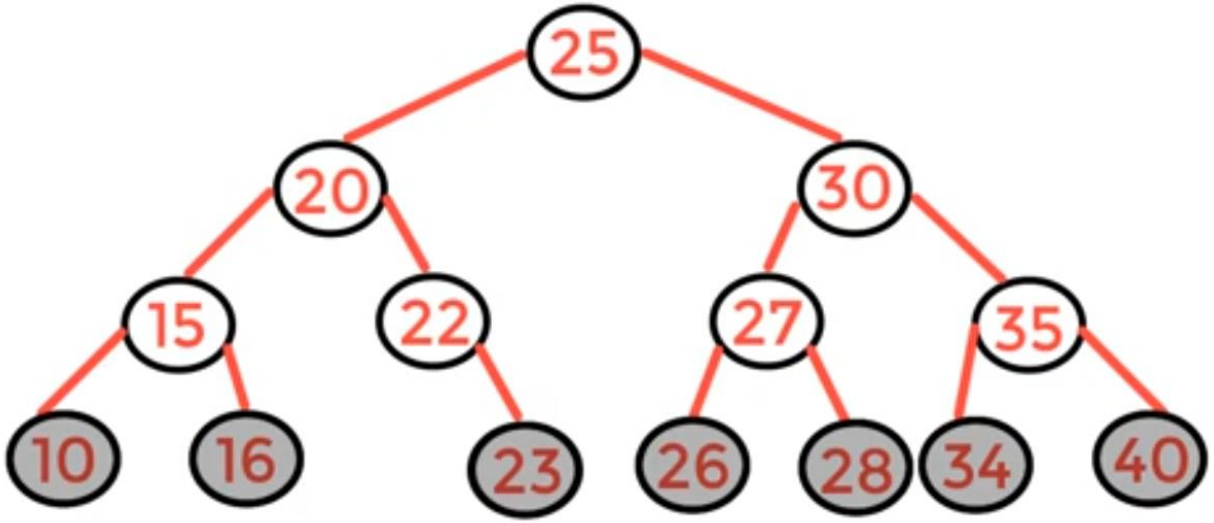
Dr. Asmaa Shaar

فهرس المحاضرة

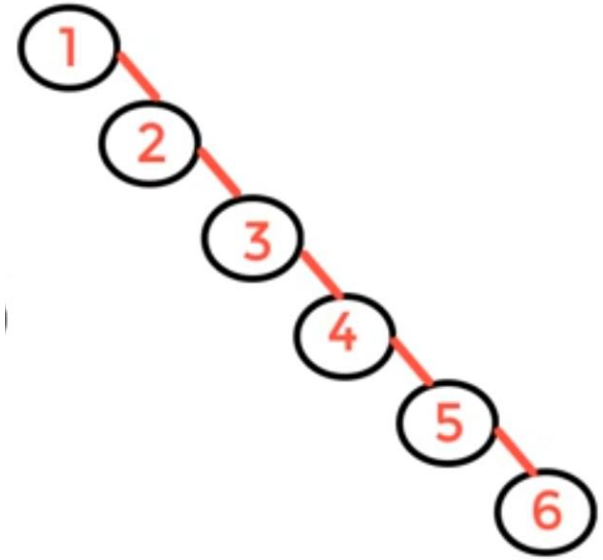
- مفهوم أشجار AVL
- دورانات شجرة AVL
- الاضافة في أشجار AVL
- الحذف من أشجار AVL

مقدمة

- تناولنا في المحاضرة السابقة **كيفية بناء** شجرة البحث الثنائية (BST).
- يتم إنجاز **عملية البحث ومعظم العمليات الاخرى** في شجرة البحث الثنائية بزمن قدره **O(h)** بحيث:
  - إذا تم بناء الشجرة بشكل جيّد ← ارتفاع الشجرة البحث الثنائية **أصغر ما يمكن** ← يكون البحث سريعاً (best case).
  - إذا كانت مجموعة البيانات مرتبة ← تكون شجرة البحث الثنائية الموافقة **خطية** ← ارتفاع الشجرة البحث الثنائية **أكبر ما يمكن** ← وبالتالي لن تكون خوارزمية البحث ذات كفاءة (worst case).

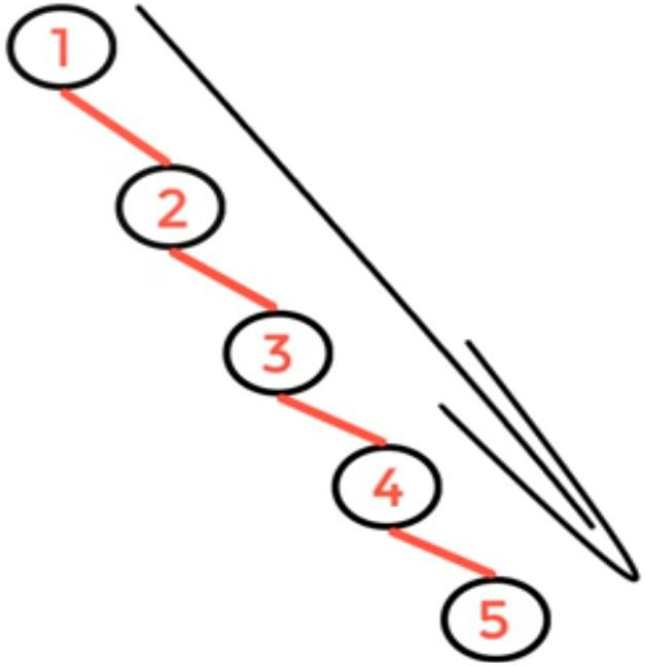


Time Complexity : $O(\log_2 n)$

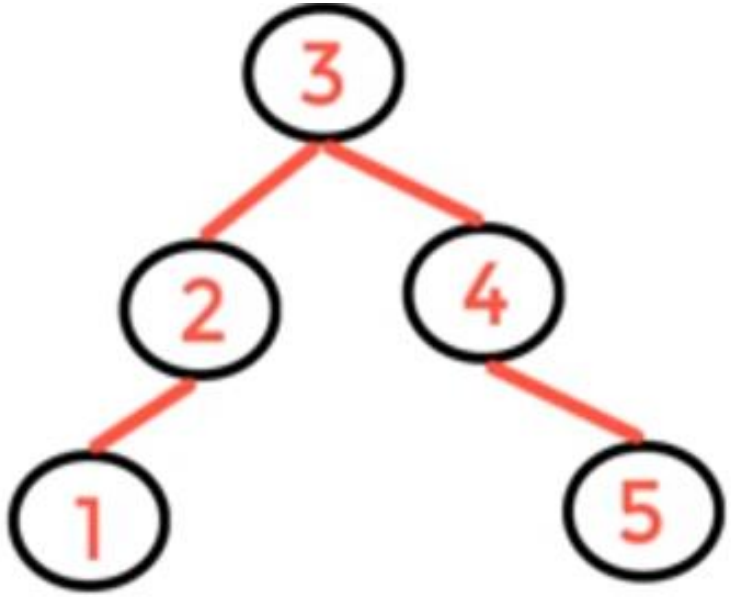


$O(\log n) \rightarrow O(n)$

مقدمة



Lack of "balance
بمقارنة ارتفاع الشجرة الفرعية اليسارية و الفرعية اليمينية نلاحظ ضياع التوازن.

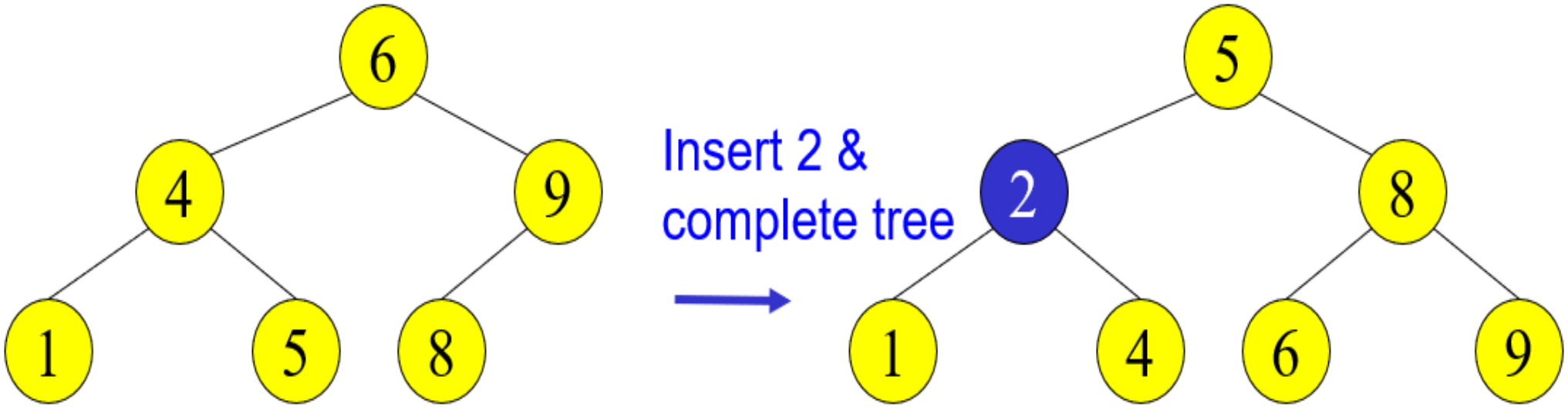


Rebalance or rotation
to bound the height of the tree to log n

- هناك عدة خوارزميات للحفاظ على أشجار البحث الثنائية **متوازنة تقريباً** بهدف تحسين كفاءة العمليات المطبقة عليها :
  - Adelson-Velskii and Landis (AVL) trees.
  - B-trees and other multiway search trees

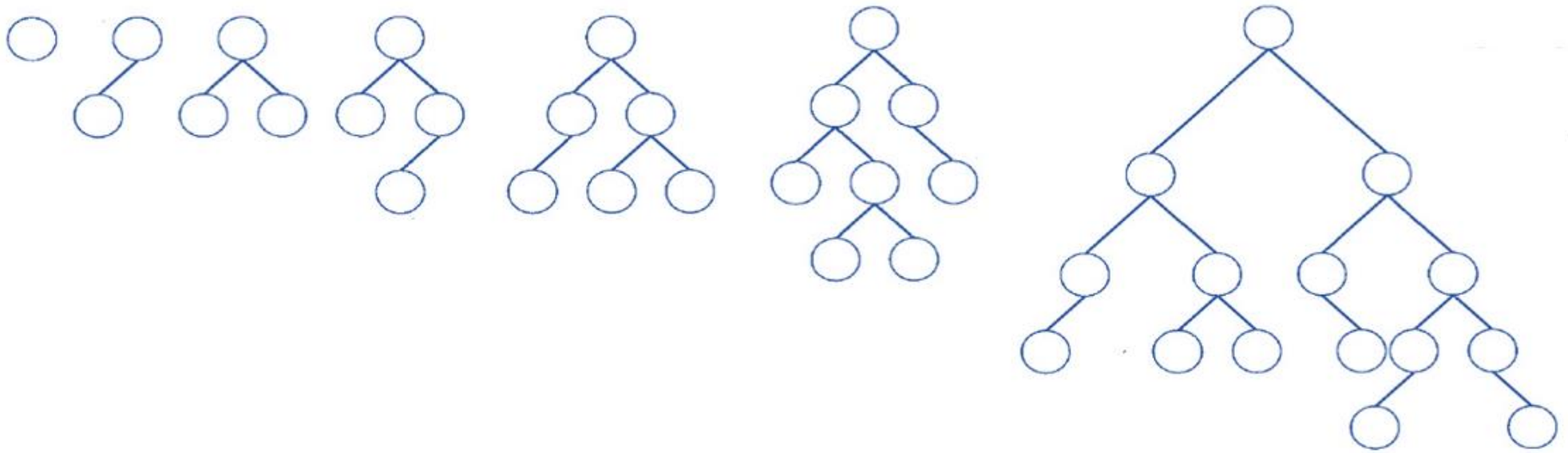
التوازن الكامل

- بعد كل عملية إضافة/حذف في شجرة البحث الثنائية، نريد الشجرة **كاملة** (complete tree).
  - شجرة ثنائية جميع مستوياتها ممتلئة بالعقد ماعدا المستوي الأخير من جهة اليمين فقط.
- يعتبر شرط **التوازن الكامل** مكلف نوعاً ما.
- مثال:



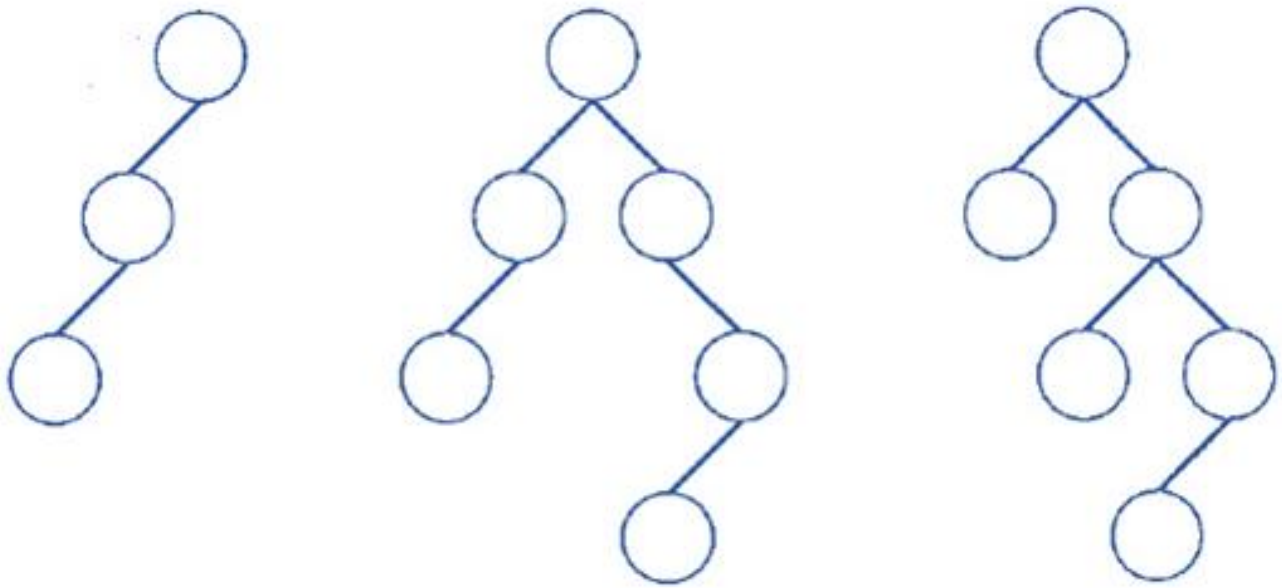
شجرة AVL

- عبارة عن **شجرة بحث ثنائية متوازنة** تحقّق:
  - يختلف ارتفاع الأشجار الفرعية اليسارية واليمينية للجذر **بمقدار 1 على الأكثر**.
  - الأشجار الفرعية اليسارية واليمينية للجذر عبارة عن أشجار AVL
- وبالتالي من أجل كل عقد في شجرة AVL، يختلف ارتفاع الأشجار الفرعية اليسارية واليمينية **بمقدار واحد على الأكثر**.
- مثال: يوضح الشكل التالي أمثلة عن أشجار من نوع AVL



شجرة AVL

- مثال: الشكل التالي يوضح أمثلة عن **أشجار ليست** من النوع AVL .

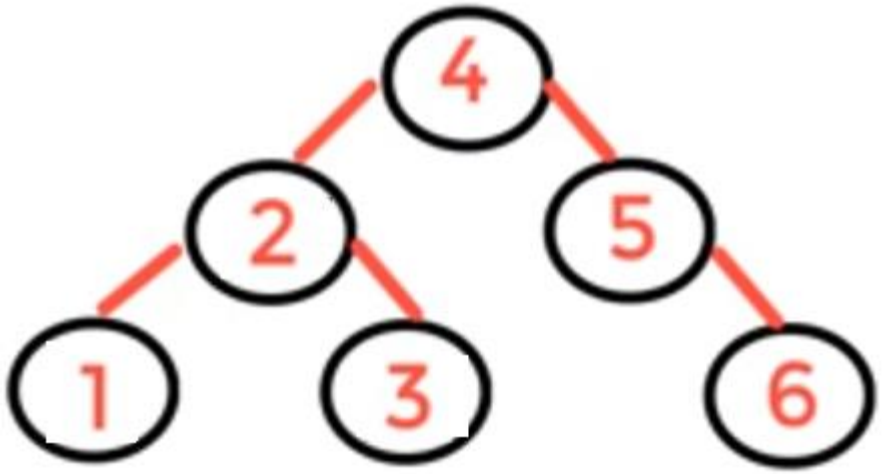


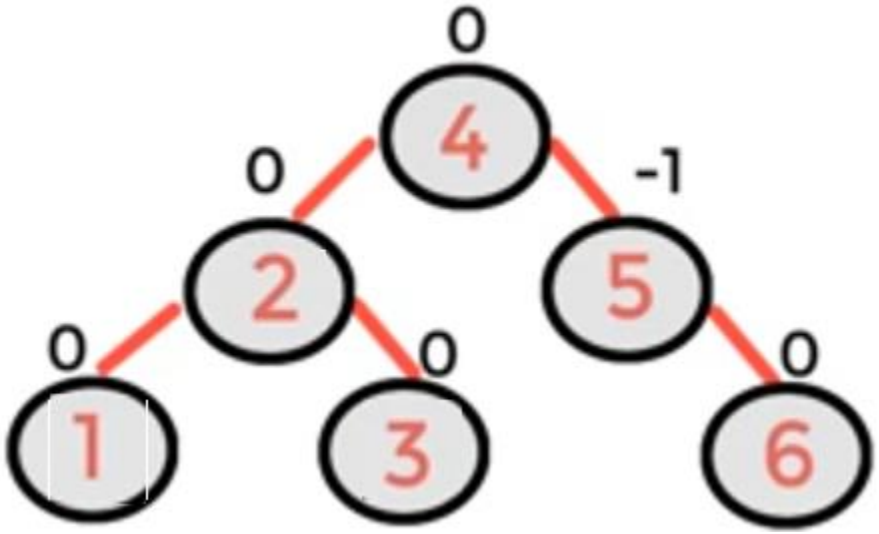
معامل التوازن

- نسمي معامل التوازن (Balance factor) لعقدة المقدار المعرف كما يلي:

BalanceFactor(empty_tree) = 0
BalanceFactor(node) = Height(LeftSubtree) - Height(RightSubtree)

- في أشجار AVL يكون معامل التوازن لجميع عقدها 1- , 0 , 1+.





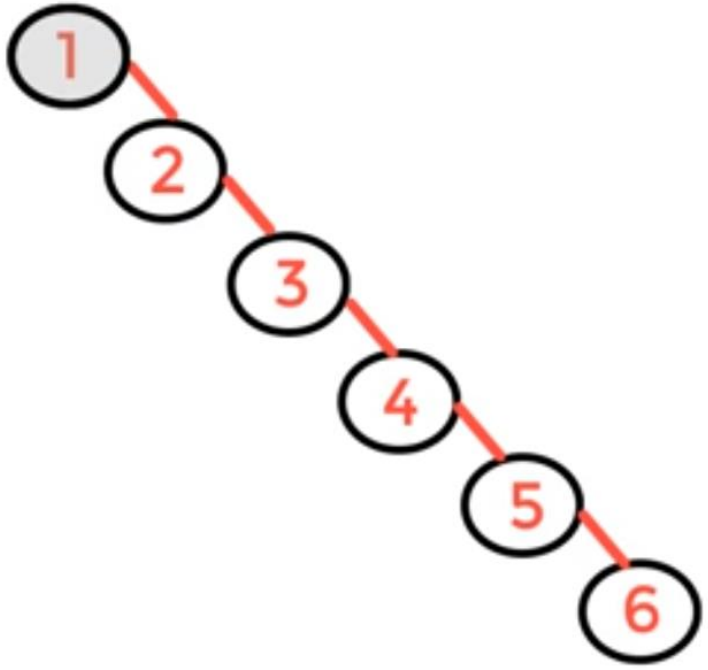
شجرة AVL

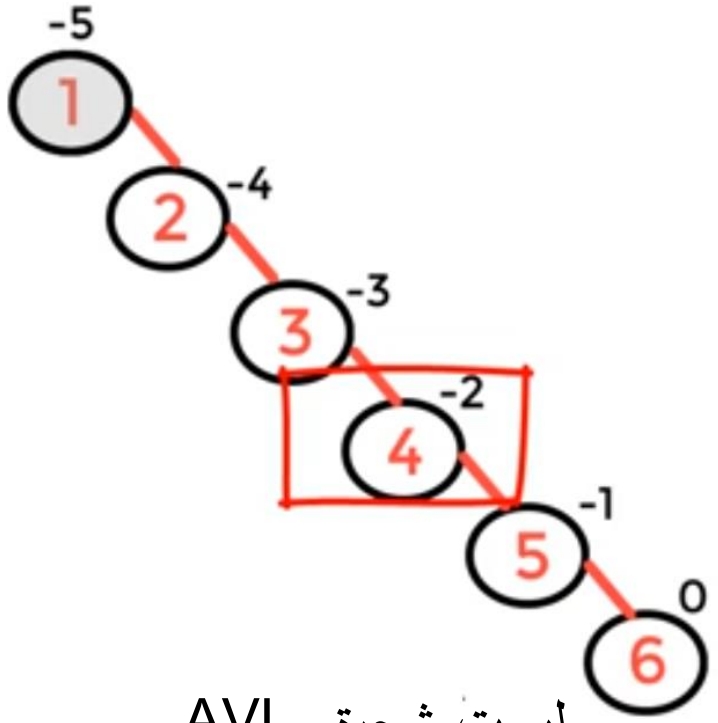
معامل التوازن

- نسمي معامل التوازن (Balance factor) لعقدة المقدار المعرف كما يلي:

BalanceFactor(empty_tree) = 0
BalanceFactor(node) = Height(LeftSubtree) - Height(RightSubtree)

- في أشجار AVL يكون معامل التوازن لجميع عقدها 1- , 0 , 1+.





ليست شجرة AVL

بنية العقدة في شجرة AVL

- بالإضافة إلى البيانات (key) والمؤشرات إلى الشجرتين الفرعيتين اليسارية واليمينية، يجب أن نقوم بتتبّع عامل توازن كل عقدة لذلك نقوم بتخزين ارتفاع العقدة (أو عامل توازن العقدة ذاتها).

```
// An AVL tree node
struct Node
{    int key;
     Node *left;
     Node *right;
     int height;};

// Get Balance factor of node N
int getBalance(Node *N)
{    if (N == NULL)          return 0;
     return height(N->left) - height(N->right);}
int height(Node *N)
{if (N == NULL)              return 0;
     return N->height;}
```

العمليات المطبقة على شجرة AVL

- بما أنّ الشجرة AVL شجرة بحث ثنائية تكون **خوارزمية البحث** لشجرة AVL هي **نفس خوارزمية البحث** لشجرة البحث الثنائية ويمكن تطبيق عمليات أخرى مثل التحقق ما إذا كانت الشجرة فارغة، و التنّقل عبر الشجرة وغيرها من العمليات بنفس الطريقة التي يتم تطبيقها على الأشجار الثنائية .
- عمليات الاضافة والحذف على أشجار AVL تكون **مختلفةً نوعاً ما**. بحيث إذا اختل شرط التوازن في شجرة AVL(بعد إضافة/حذف عقدة) فإن **إعادة التوازن** تجري **بتحويلات موضعية بسيطة** على بعض عقد الشجرة ... تسمى هذه العمليات بالدورانات

الدوران هو نقل لبعض مكونات الشجرة من جهة إلى أخرى لإعادة توازنها مع الحفاظ على شجرة البحث الثنائية **keys(left) < key(root) < keys(right)**

- يوجد نوعان من الدورانات:
  1. دورانات بسيطة (simple rotations): دوران بسيط إلى اليسار (LR) أو دوران بسيط إلى اليمين (RR).
  2. دورانات مزدوجة (double rotations): دوران إلى اليسار ثم إلى اليمين (LRR) أو دوران إلى اليمين ثم إلى اليسار (RLR).

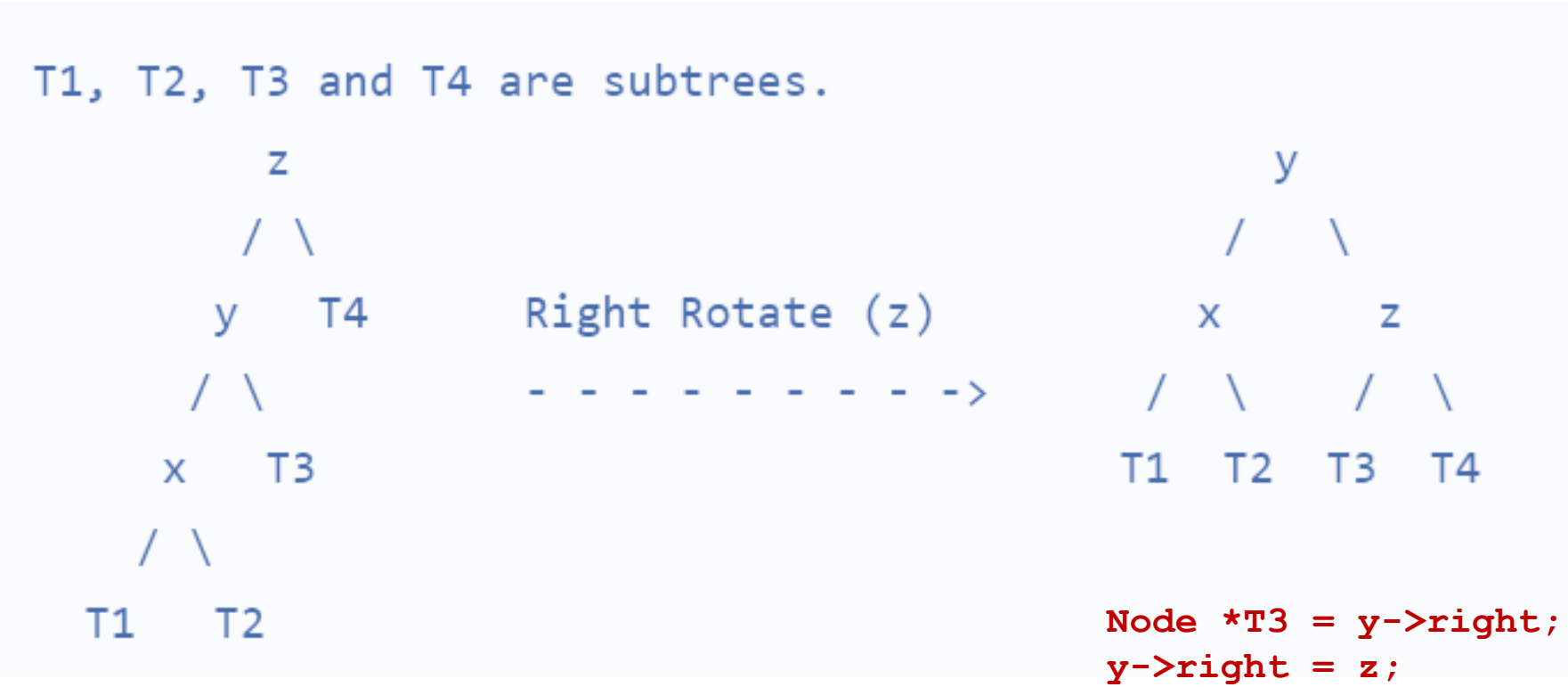
الإضافة في شجرة AVL

الإضافة في شجرة AVL

- بفرض العقدة المراد اضافتها هي w، تتم الإضافة وفق الخطوات التالية:
  1. يتم تنفيذ **عملية الاضافة نفسها** في BST
  2. بعد إضافة العقدة الجديدة w **قد لا تكون الشجرة الناتجة** شجرة AVL. لذلك يجب أن نستعيد توازن الشجرة وذلك عن طريق زيارة العقد الموجودة على نفس المسار بدءاً من العقدة المضافة w صعوداً لإيجاد **العقدة الأولى غير المتوازنة** .
     - هنا يتم التركيز على ثلاثة عقد:
       1. z هي العقدة الأولى غير المتوازنة (first unbalanced node).
       2. y هي العقدة الابن لـ z والتي تكون ضمن المسار من w لـ z
       3. x هي حفيد لـ z التي تأتي ضمن المسار من w لـ z.
  3. يتم إعادة توازن الشجرة من خلال تنفيذ **الدورانات المناسبة** على الشجرة الفرعية التي جذرها z.
     - هناك اربع حالات ممكنة يجب معالجتها، حسب ترتيب y, x, z

الحالة الأولى: Left Left Case

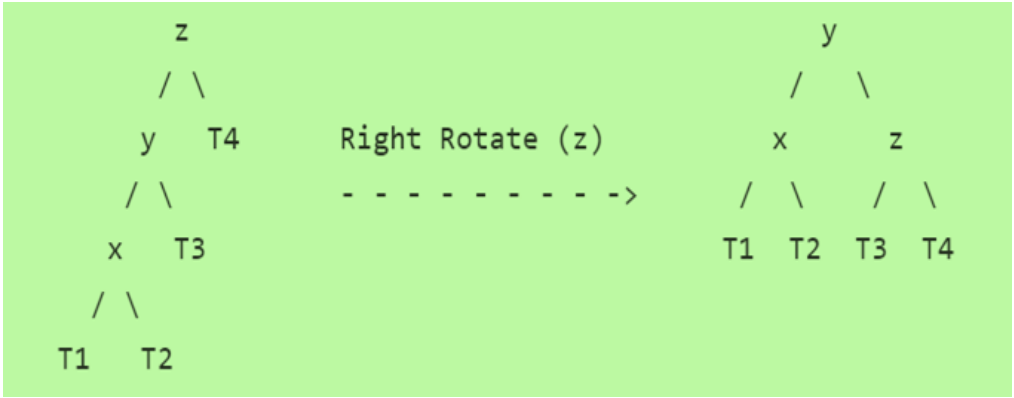
- ارتفاع الأشجار الفرعية T1, T2, T3, T4 متساوي و ليكن h .
- Y هي الابن اليساري لـ z و x هي الابن اليساري لـ y ← دوران وحيد لليمين عند z.




```
Node *T3 = y->right;
y->right = z;
z->left = T3;
```

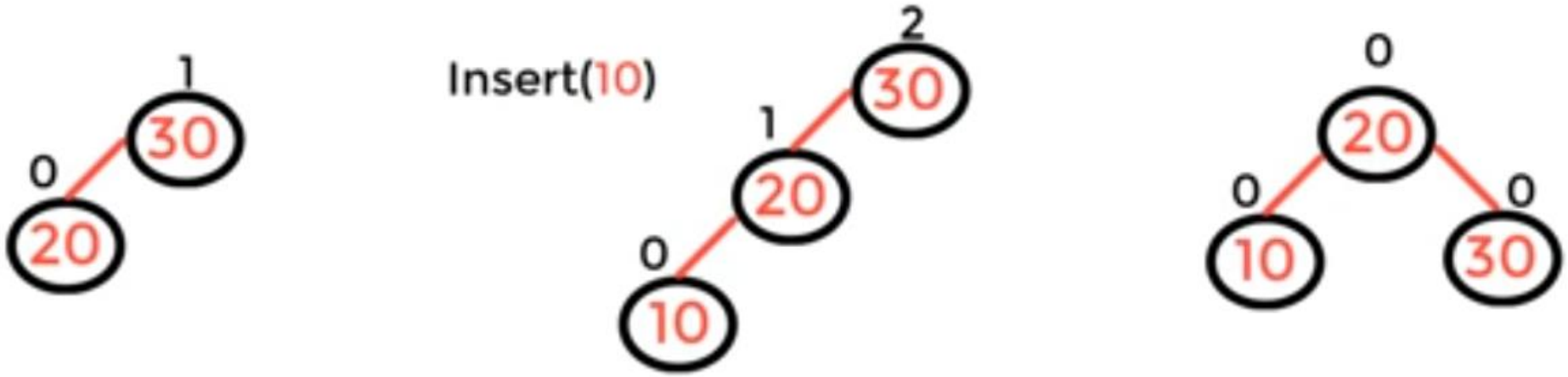
Left Left Case :الحالة الأولى

```
Node *rightRotate(Node *z)
{    Node *y = z->left;
     Node *T3 = y->right;
     // Perform rotation
     y->right = z;
     z->left = T3;
     // Update heights
     z->height = max(height(z->left),
                     height(z->right)) + 1;
     y->height = max(height(y->left),
                     height(y->right)) + 1;
     // Return new root
     return y;
}
```

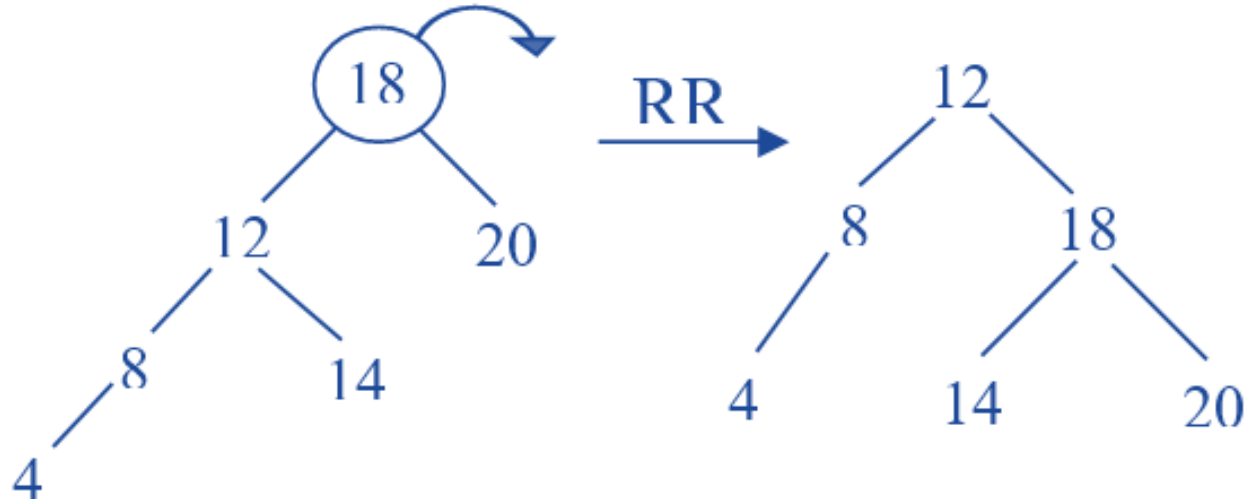


الحالة الأولى: Left Left Case

- مثال 1:

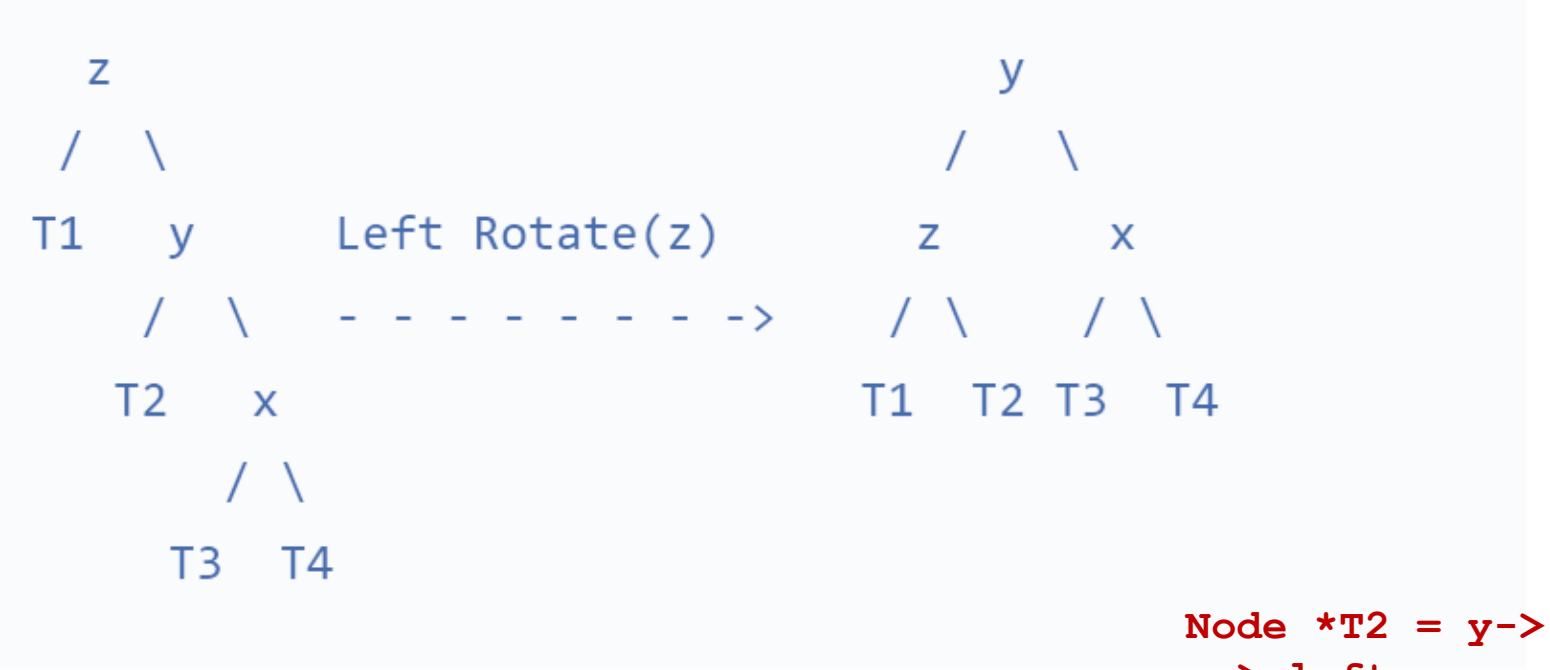


- مثال 2:



الحالة الثانية: Right Right Case

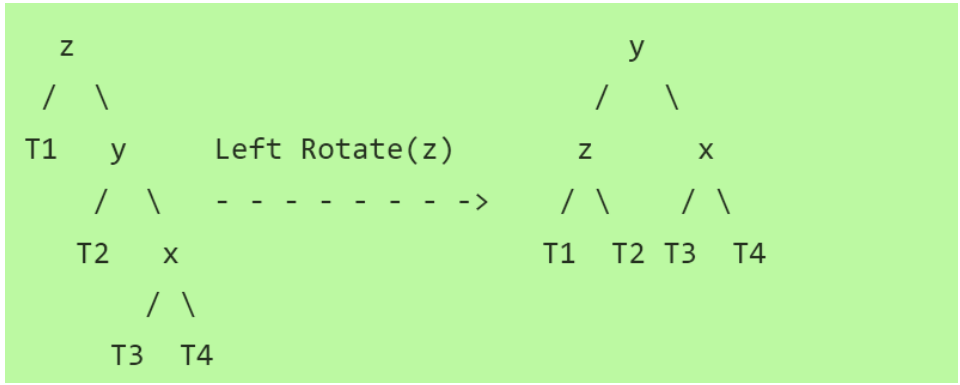
- ارتفاع الأشجار الفرعية T1, T2, T3, T4 متساوي و ليكن h .
- Y هي الابن اليميني لـ z و x هي الابن اليميني لـ y ← دوران وحيد لليسار عند z.




```
Node *T2 = y-> left;
y-> left = z;
z-> right = T2;
```

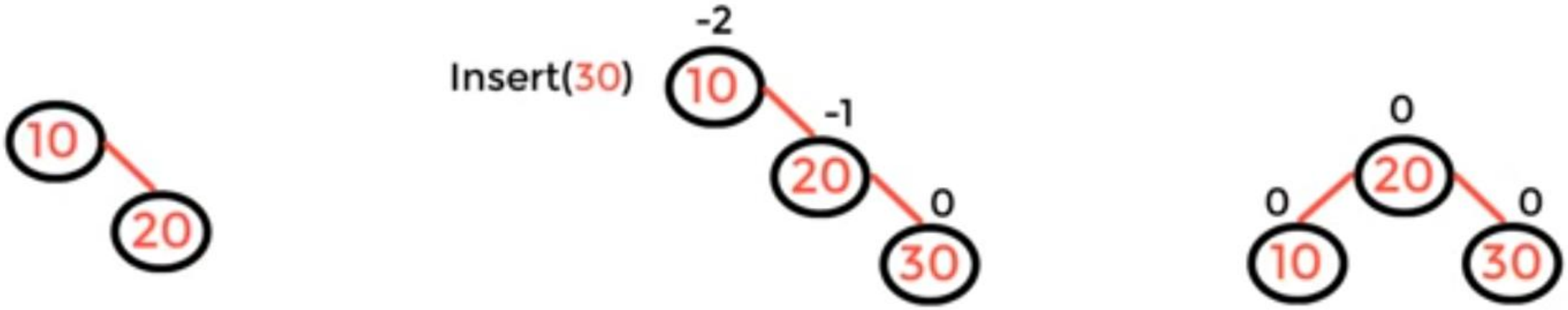
Right Right Case :الحالة الثانية

```
Node *leftRotate(Node *z)
{
    Node *y = z->right;
    Node *T2 = y->left;
    // Perform rotation
    y->left = z;
    z->right = T2;
    // Update heights
    z->height = max(height(z->left),
                    height(z->right)) + 1;
    y->height = max(height(y->left),
                    height(y->right)) + 1;
    // Return new root
    return y;
}
```

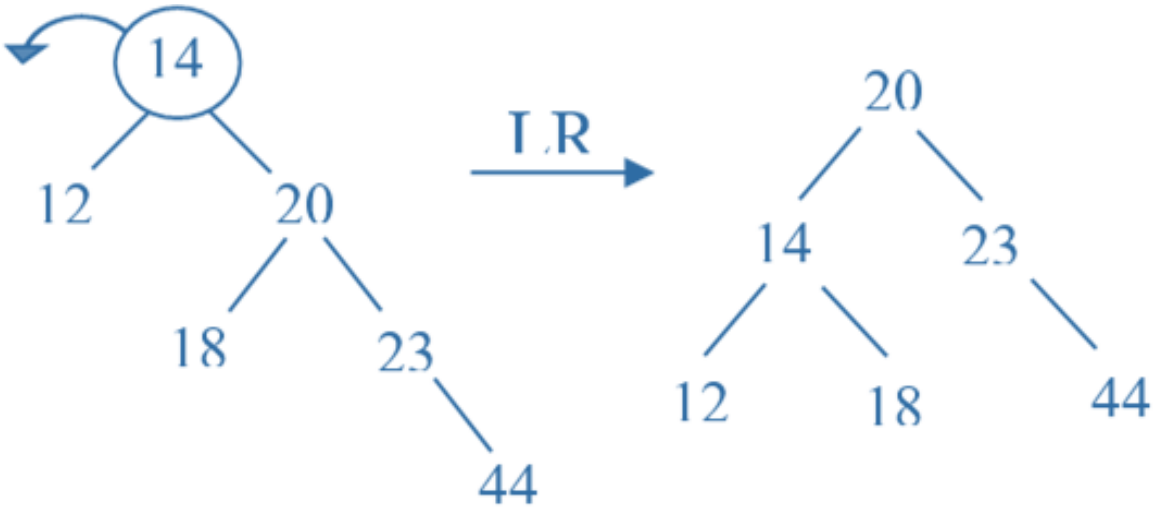


Right Right Case :الحالة الثانية

- مثال 1:

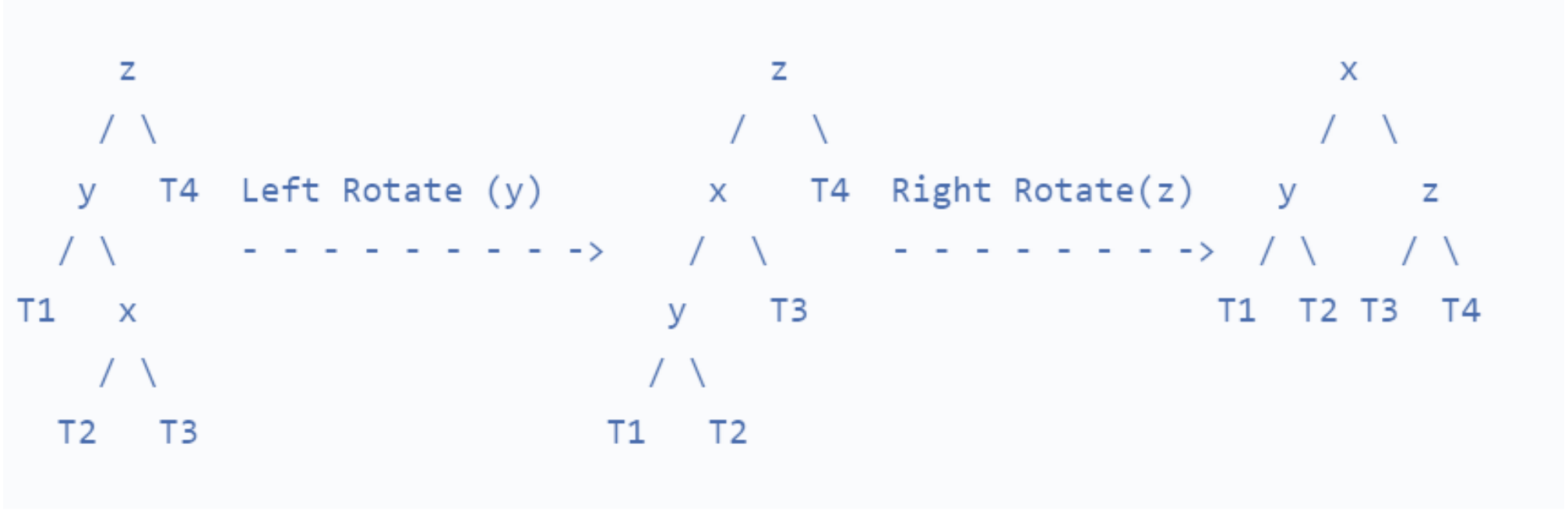


- مثال 2:



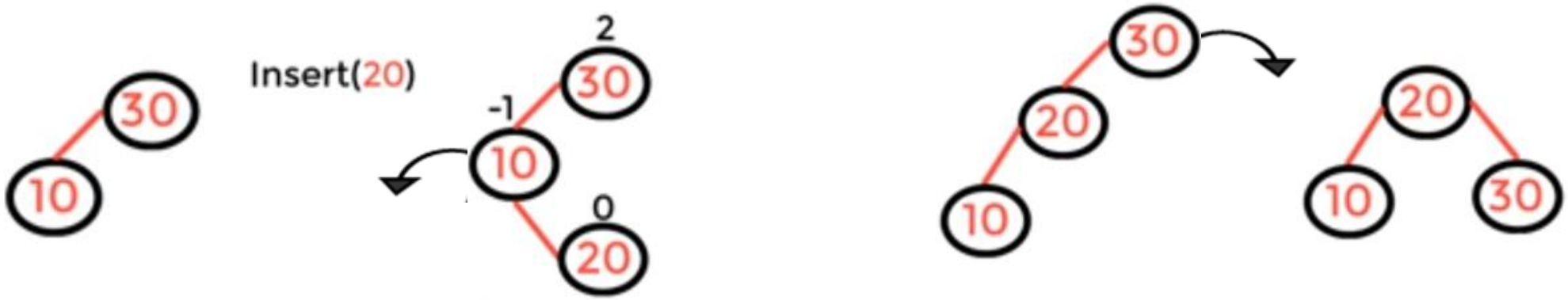
الحالة الثالثة: Left Right Case

- ارتفاع الأشجار الفرعية T1, T2, T3, T4 متساوي و ليكن h .
- Y هي الابن اليساري لـ z و x هي الابن اليميني لـ y ← دوران مزدوج يساري عند y– يميني عند z.

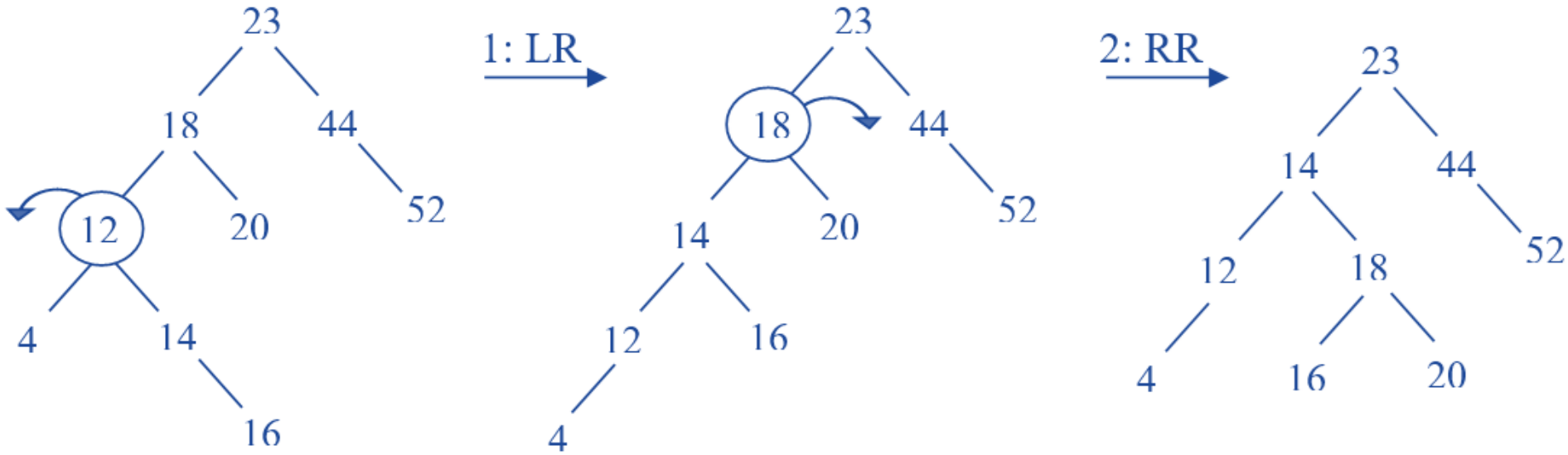


الحالة الثالثة: Left Right Case

- مثال 1:

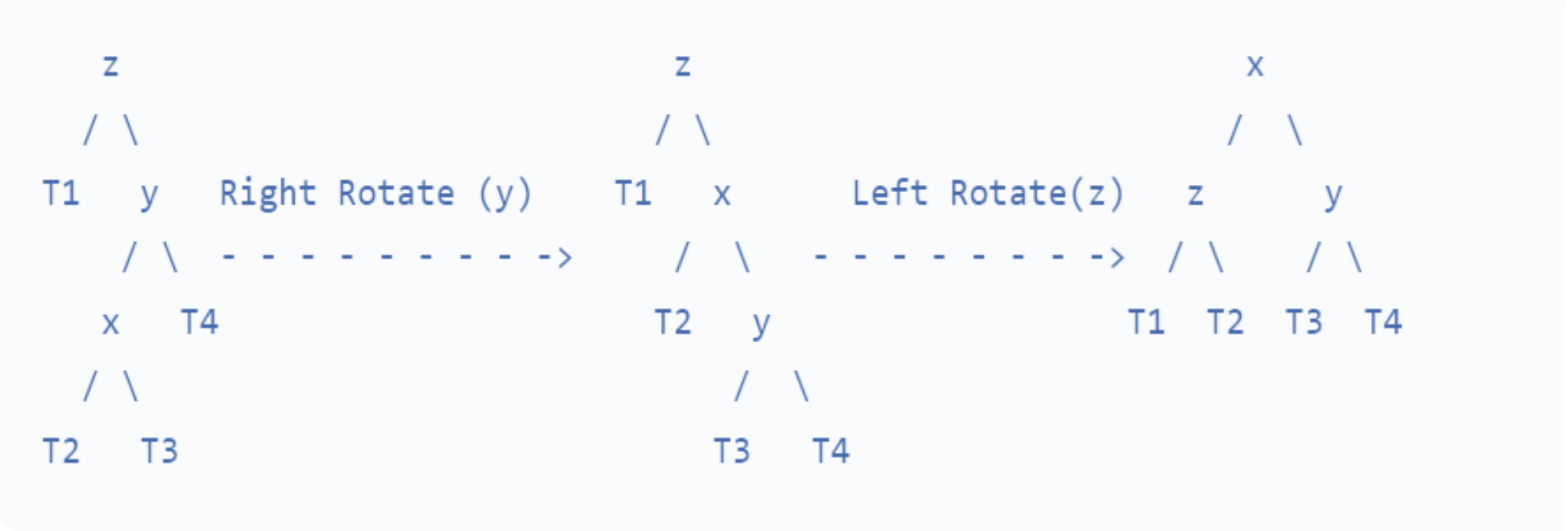


- مثال2:



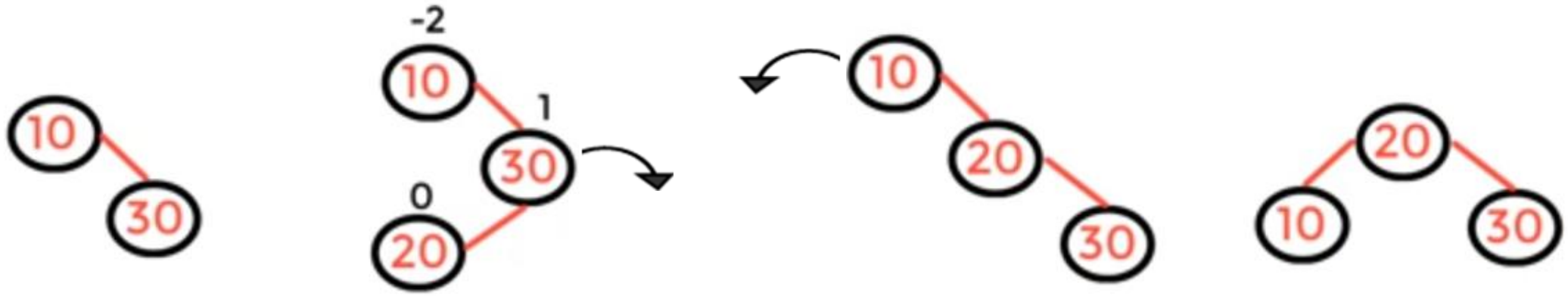
الحالة الرابعة: Right Left Case

- ارتفاع الأشجار الفرعية T1, T2, T3, T4 متساوي و ليكن h .
- Y هي الابن اليميني لـ z و x هي الابن اليساري لـ y ← دوران مزدوج يميني عند y -يساري عند z.

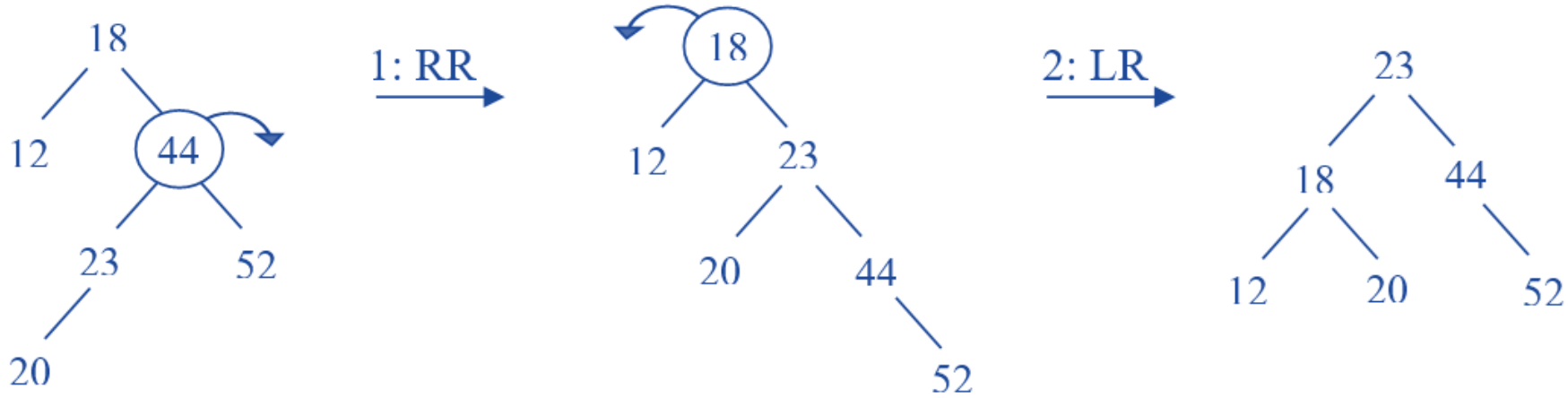


Right Left Case :الحالة الرابعة

- مثال 1:



- مثال2:



الإضافة في شجرة AVL

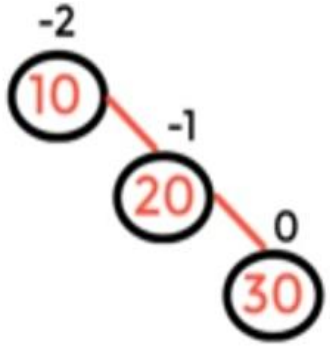
```
// Recursive function to insert a key
// in the subtree rooted with node and
// returns the new root of the subtree.
Node* insert(Node* node, int key)
{
    /* 1. Perform the normal BST insertion */
    if (node == NULL)
        return(newNode(key));

    if (key < node->key)
        node->left = insert(node->left, key);
    else if (key > node->key)
        node->right = insert(node->right, key);
    else // Equal keys are not allowed in BST
        return node;

    /* 2. Update height of this ancestor node */
    node->height = 1 + max(height(node->left),
                           height(node->right));
```

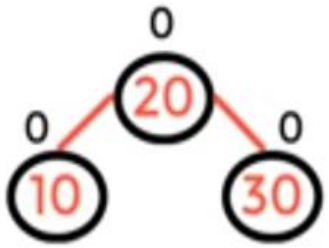
الإضافة في شجرة AVL

```
/* 3. Get the balance factor of this ancestor
    node to check whether this node became unbalanced */
int balance = getBalance(node);
// If this node becomes unbalanced, then there are 4 cases
// Left Left Case
if (balance > 1 && key < node->left->key)
    return rightRotate(node);
// Right Right Case
if (balance < -1 && key > node->right->key)
    return leftRotate(node);
// Left Right Case
if (balance > 1 && key > node->left->key)
{   node->left = leftRotate(node->left);
    return rightRotate(node);}
// Right Left Case
if (balance < -1 && key < node->right->key)
{   node->right = rightRotate(node->right);
    return leftRotate(node);}
/* return the (unchanged) node pointer */
return node;}
```

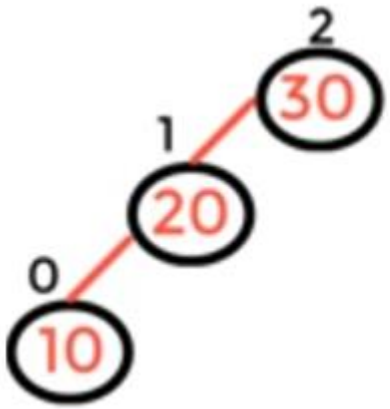
الدوران البسيطة

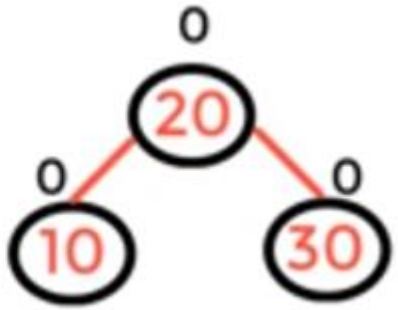
- **ملاحظة:** إذا كان عامل توازن العقدة z (التي يتم عندها إعادة توازن الشجرة) وعامل توازن العقدة الابن y (الموجودة على مسار العقدة المحشورة) لهما **نفس الاشارة** عندئذ نقوم بعمل **دوران واحد** عند z.
  - إذا كانت عوامل التوازن **سالبة** عندئذ نقوم بعمل دوران واحد لليسار عند z .





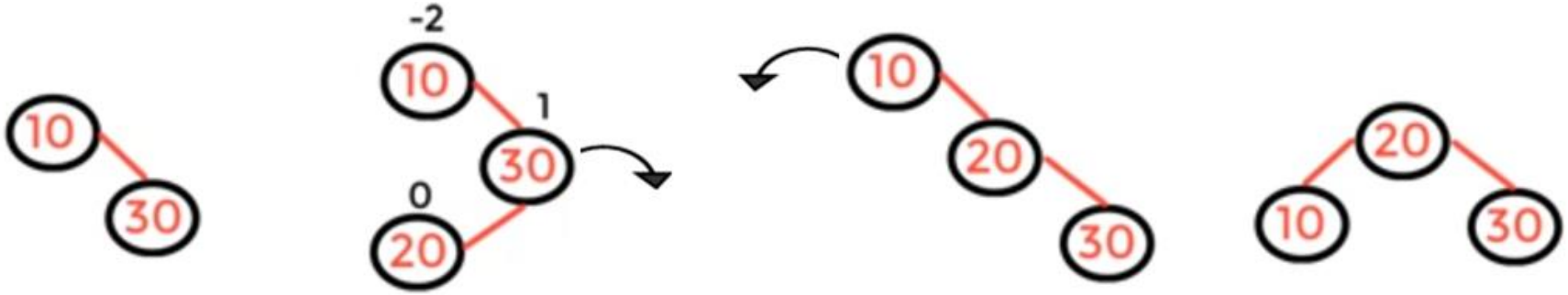
  - إذا كانت عوامل التوازن **موجبة** عندئذ نقوم بعمل دوران واحد لليمين عند z .



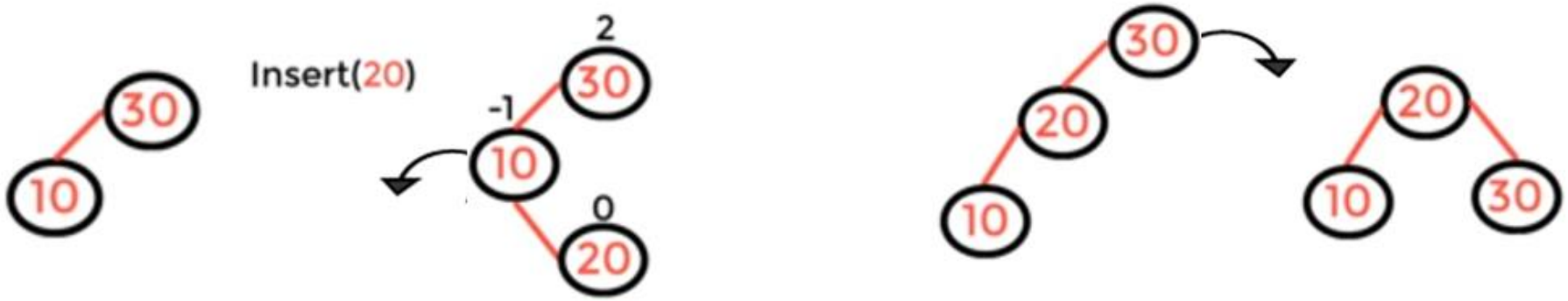


الدوران المزدوج

- **ملاحظة:** إذا كان عامل توازن العقدة z (التي يتم عندها إعادة توازن الشجرة) وعامل توازن العقدة الابن y (الموجودة على مسار العقدة المحشورة) **متعاكسين بالإشارة** عندئذ نقوم بعمل **دوران مزدوج.**
  - إذا كانت عامل توازن العقدة z **سالب** عندئذ نقوم بعمل **دوران مزدوج يمين عند y– يسار عند z.**

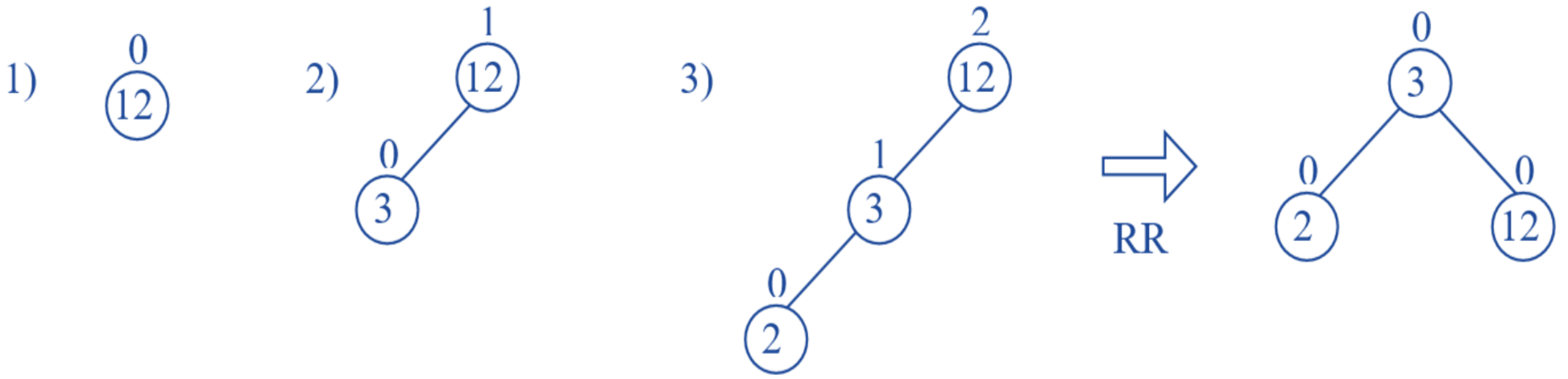


إذا كانت عامل توازن العقدة z **موجب** عندئذ نقوم بعمل **دوران مزدوج يسار عند y– يمين عند z.**



الإضافة في أشجار AVL

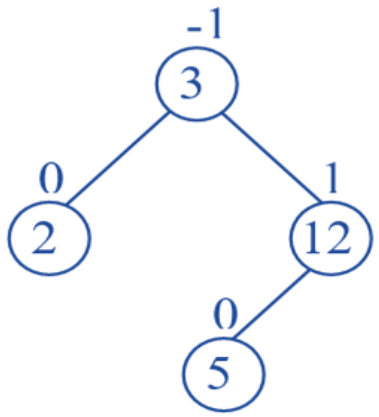
- **مثال 1:** قم ببناء شجرة AVL لكل من العقد التالية خطوة بخطوة من خلال حشر كل عقدة ومن ثم إعادة موازنة الشجرة اذا استدعى الامر. { 12,3,2,5,4,7,9,11,14,10}



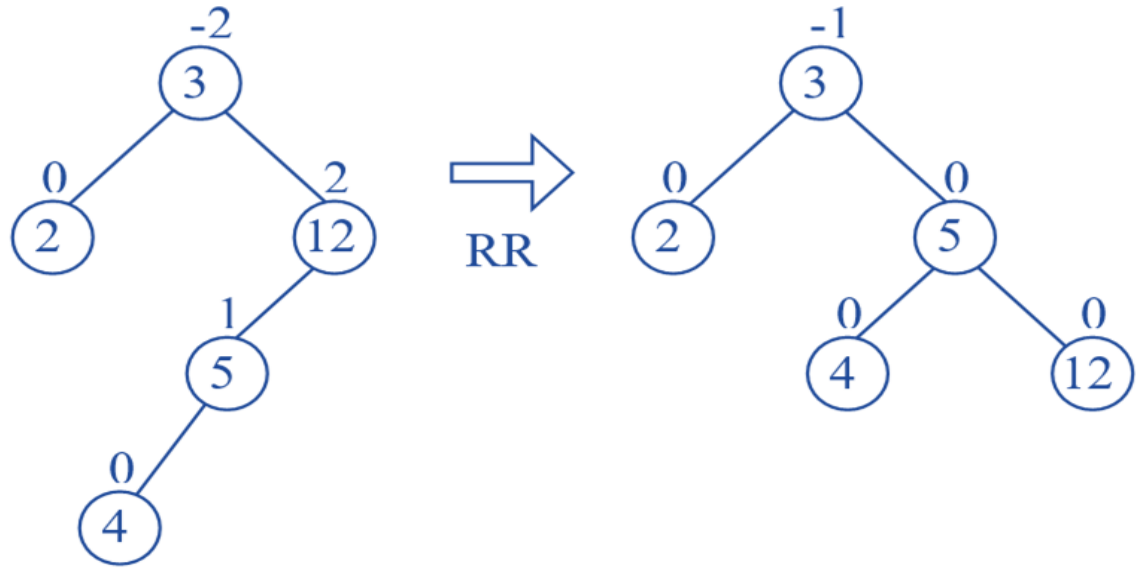
الإضافة في أشجار AVL

- **مثال 1:** قم ببناء شجرة AVL لكل من العقد التالية خطوة بخطوة من خلال حشر كل عقدة كورقة ومن ثم إعادة موازنة الشجرة اذا استدعى الامر. {12,3,2,**5,4**,7,9,11,14,10}

4)

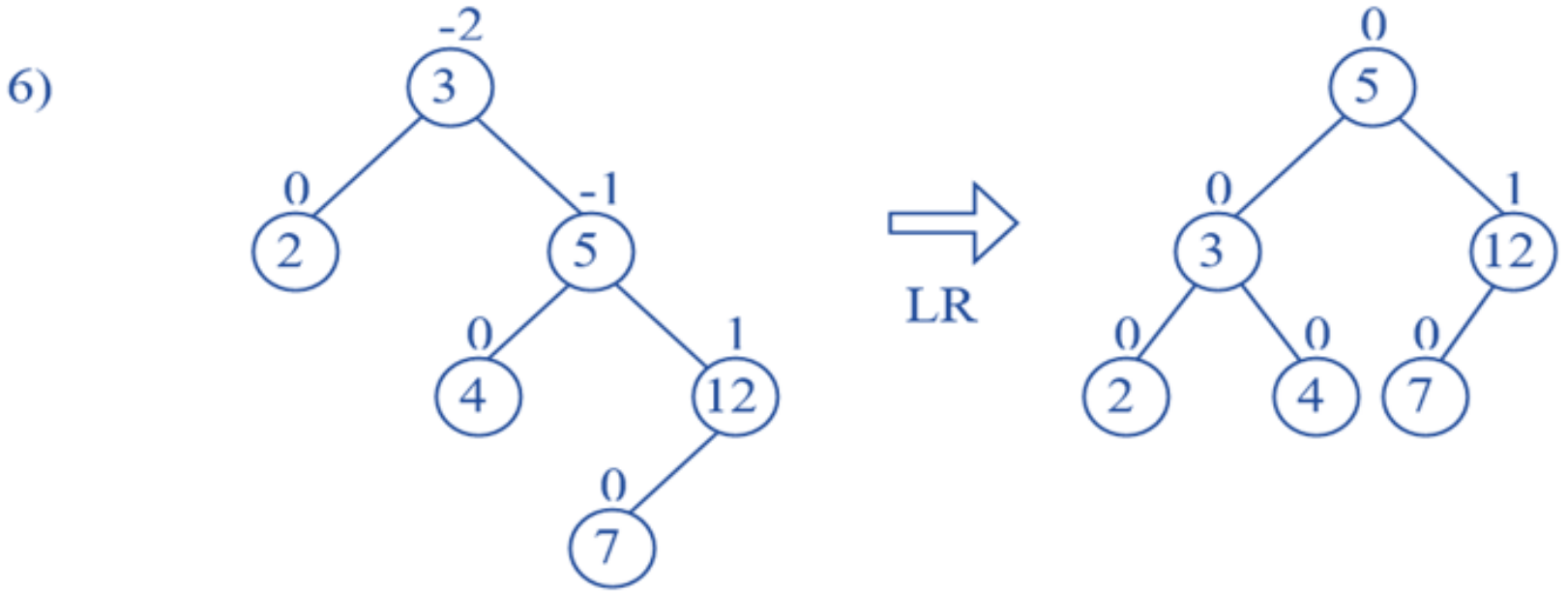


5)



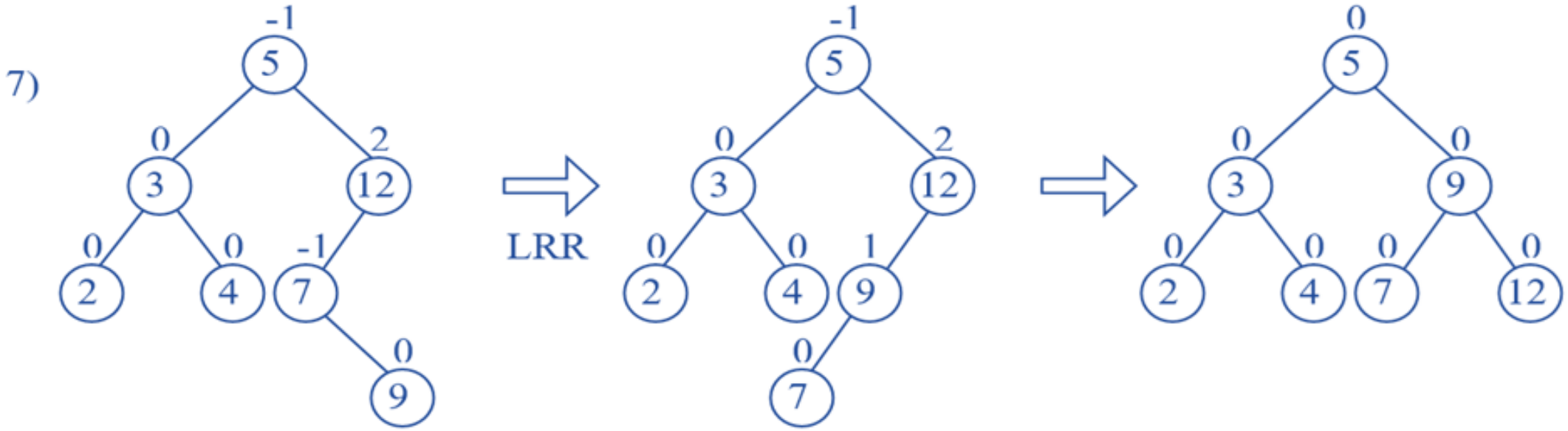
الإضافة في أشجار AVL

- **مثال 1:** قم ببناء شجرة AVL لكل من العقد التالية خطوة بخطوة من خلال حشر كل عقدة كورقة ومن ثم إعادة موازنة الشجرة اذا استدعى الامر. {12,3,2,5,4,7,9,11,14,10}



الإضافة في أشجار AVL

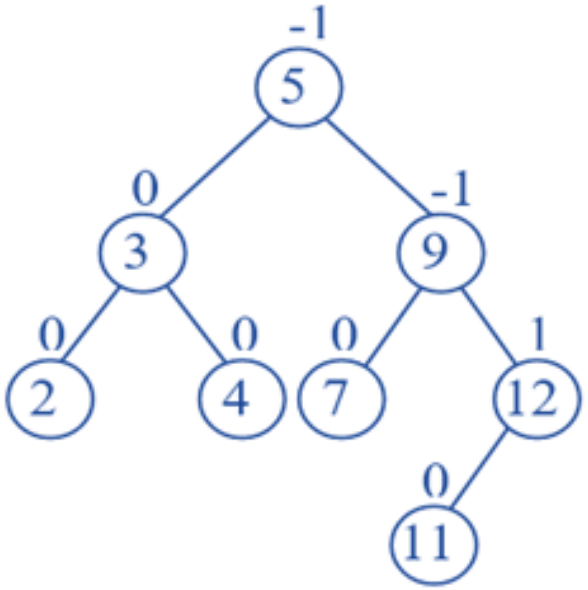
- **مثال 1:** قم ببناء شجرة AVL لكل من العقد التالية خطوة بخطوة من خلال حشر كل عقدة كورقة ومن ثم إعادة موازنة الشجرة اذا استدعى الامر. { 12,3,2,5,4,7,**9**,11,14,10}



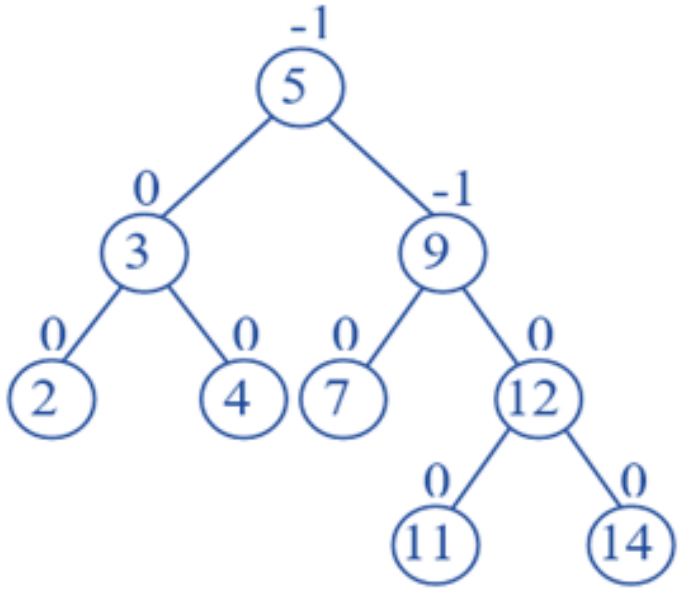
الإضافة في أشجار AVL

- **مثال 1:** قم ببناء شجرة AVL لكل من العقد التالية خطوة بخطوة من خلال حشر كل عقدة كورقة ومن ثم إعادة موازنة الشجرة اذا استدعى الامر. { 12,3,2,5,4,7,9,**11,14**,10}

8)

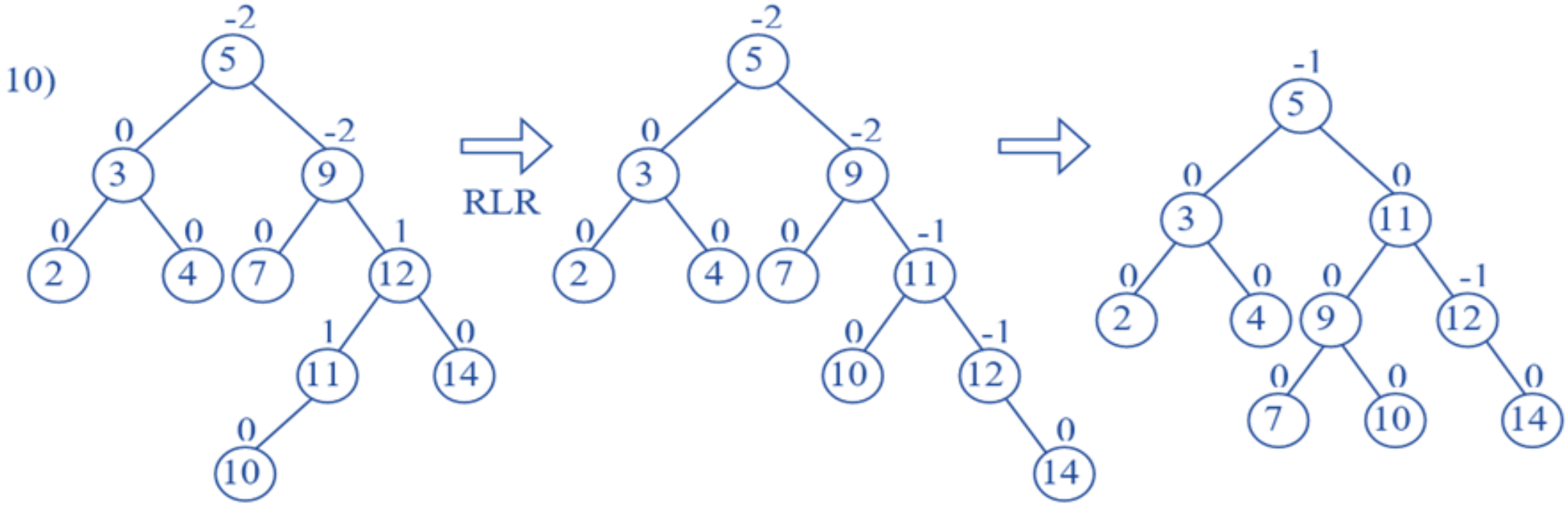


9)



الإضافة في أشجار AVL

- **مثال 1:** قم ببناء شجرة AVL لكل من العقد التالية خطوة بخطوة من خلال حشر كل عقدة كورقة ومن ثم إعادة موازنة الشجرة اذا استدعى الامر. {12,3,2,5,4,7,9,11,14,**10**}



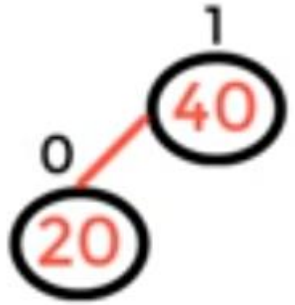
الإضافة في أشجار AVL

- مثال 2:

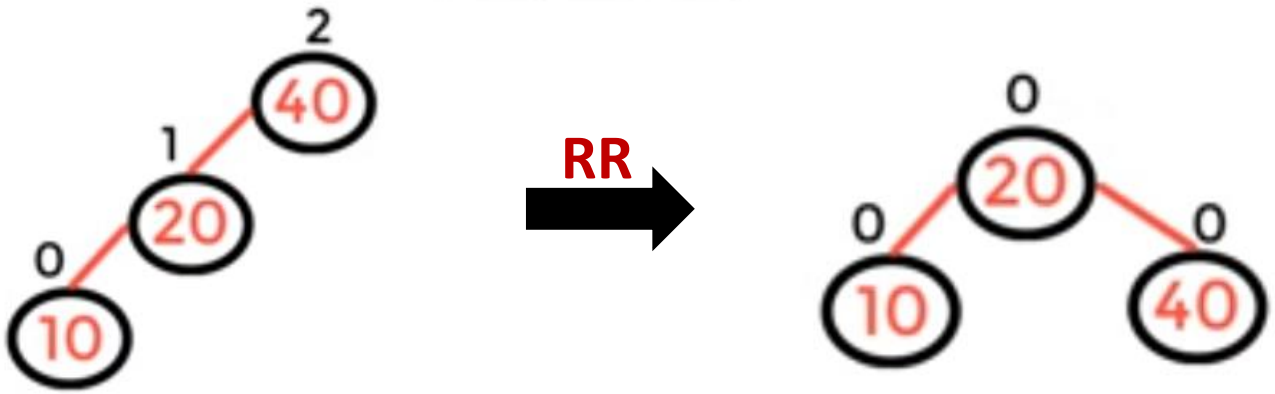
Keys: 40 , 20 , 10 , 25 , 30 , 22 , 50

Keys: 40 , 20 , 10 , 25 , 30 , 22 , 50

Keys: 40 , 20 , 10 , 25 , 30 , 22 , 50



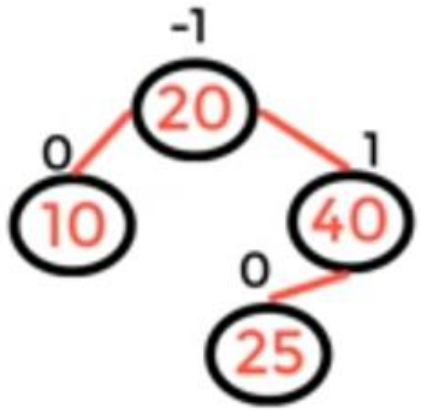
Keys: 40 , 20 , 10 , 25 , 30 , 22 , 50



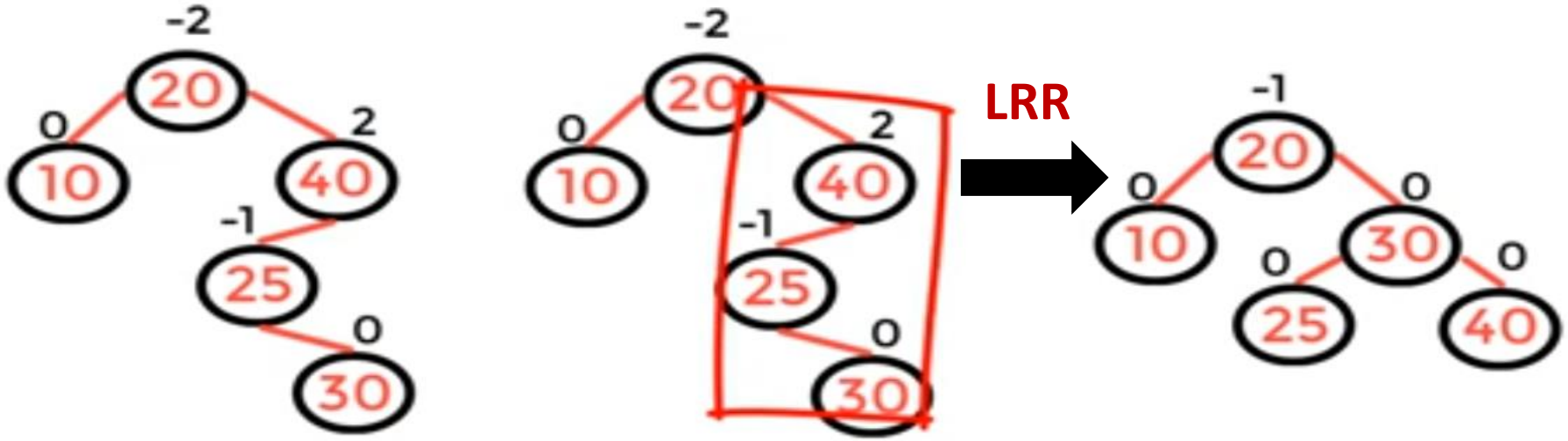
الإضافة في أشجار AVL

- مثال 2:

Keys: 40 , 20 , 10 , 25 , 30 , 22 , 50



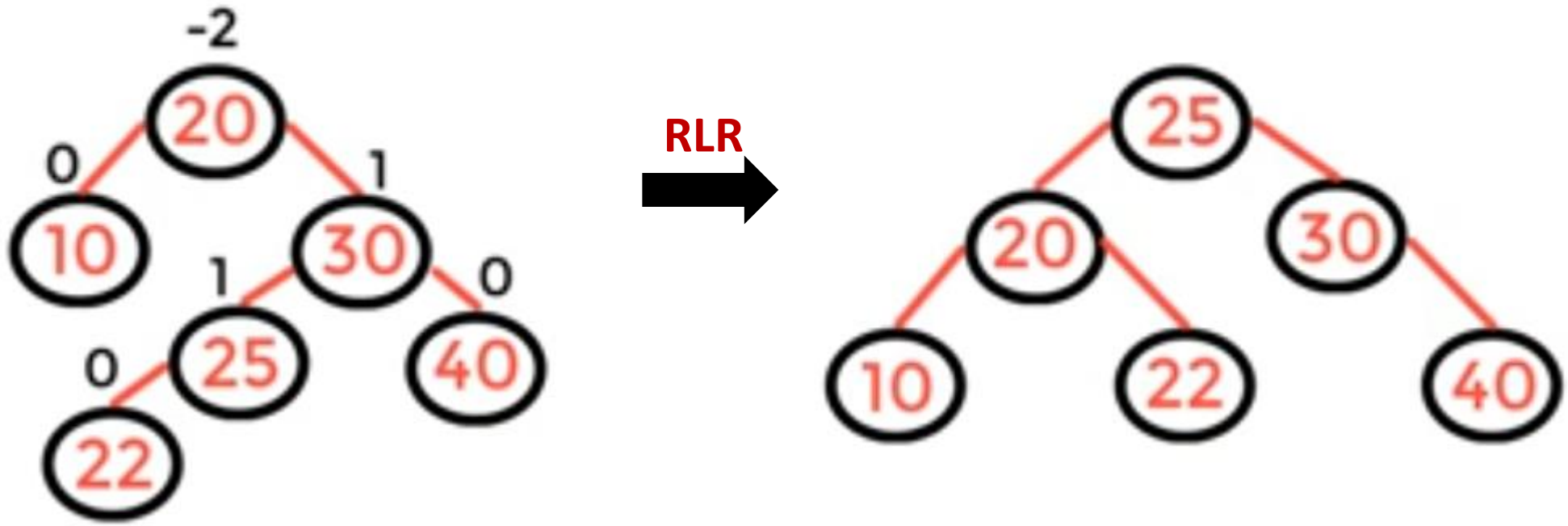
Keys: 40 , 20 , 10 , 25 , 30 , 22 , 50



الإضافة في أشجار AVL

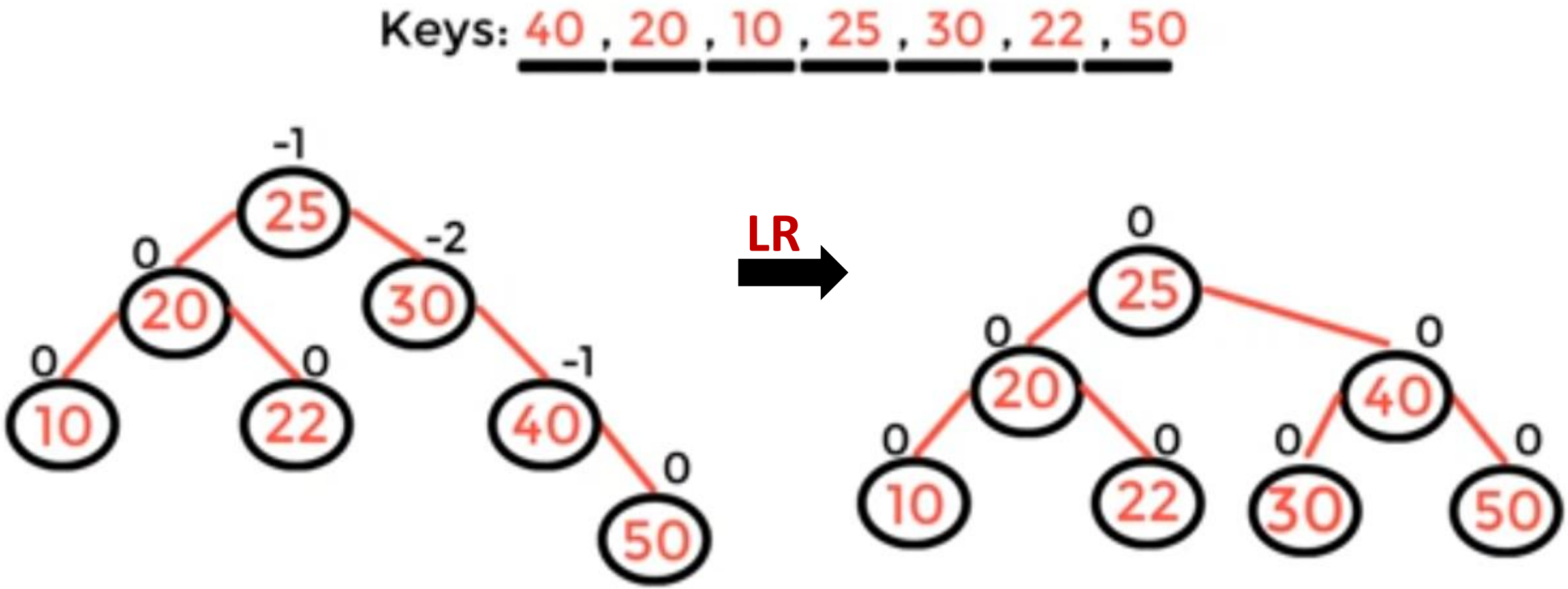
- مثال 2:

Keys: 40 , 20 , 10 , 25 , 30 , 22 , 50



الإضافة في أشجار AVL

- مثال 2:



حذف عقدة في شجرة AVL

حذف عقدة في شجرة AVL

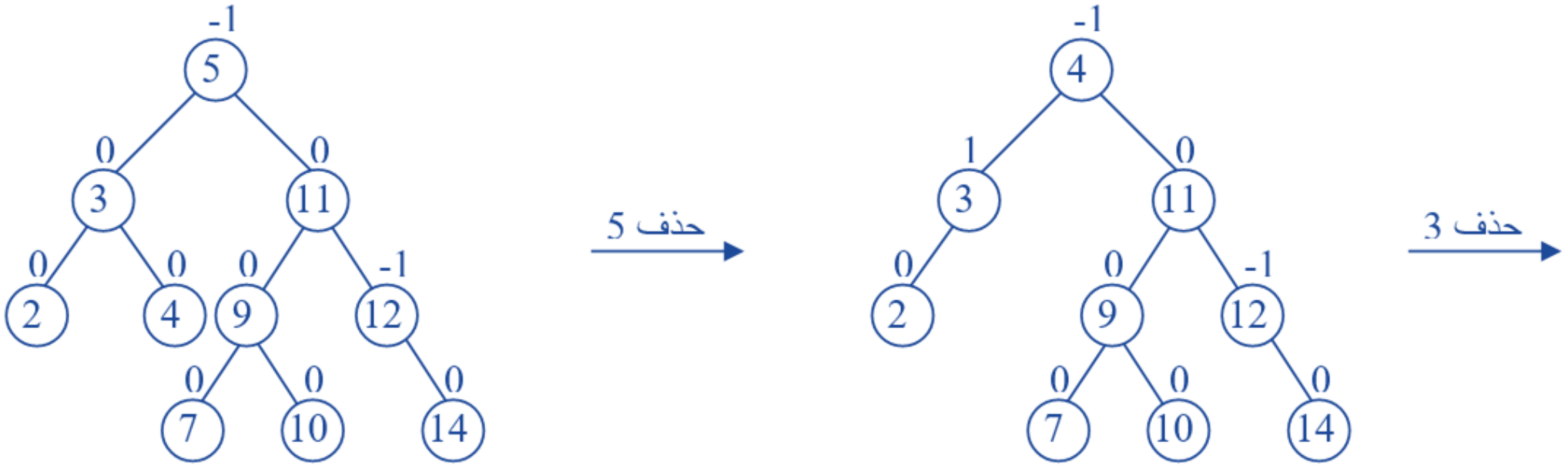
- بفرض العقدة المراد حذفها هي w، يتم الحذف وفق الخطوات التالية:
  1. يتم تنفيذ **عملية الحذف نفسها** في BST.
  2. بعد حذف العقدة قد لا تكون الشجرة الناتجة شجرة AVL. لذلك يجب أن **نستعيد توازن** الشجرة وذلك عن طريق زيارة العقد الموجودة على نفس المسار بدءاً من العقدة w صعوداً لإيجاد العقدة الأولى غير المتوازنة.
     - هنا يتم التركيز على ثلاثة عقد:
       1. z هي العقدة الأولى غير المتوازنة (first unbalanced node).
       2. y هي العقدة الابن لـ z والتي تكون **الأكثر ارتفاعاً**.
       3. x هي العقدة الابن لـ y والتي تكون **الأكثر ارتفاعاً**.
  3. يتم إعادة توازن الشجرة من خلال تنفيذ **الدورانات المناسبة** على الشجرة الفرعية التي جذرها z.
     - هناك اربع حالات ممكنة يجب معالجتها، حسب ترتيب y, x, z

حذف عقدة في شجرة AVL

```
Node* deleteNode(Node* root, int key)
 {// STEP 1: PERFORM STANDARD BST DELETE
          ...
  // STEP 2: UPDATE HEIGHT OF THE CURRENT NODE
  root->height = 1 + max(height(root->left),height(root->right));
  // STEP 3: GET THE BALANCE FACTOR OF
  // THIS NODE (to check whether this node became unbalanced)
  int balance = getBalance(root);
  // If this node becomes unbalanced,then there are 4 cases
  // Left Left Case
  if (balance > 1 && getBalance(root->left) >= 0)
      return rightRotate(root);
  // Left Right Case
  if (balance > 1 && getBalance(root->left) < 0)
  {    root->left = leftRotate(root->left);
      return rightRotate(root);}
  // Right Right Case
  if (balance < -1 && getBalance(root->right) <= 0)
      return leftRotate(root);
  // Right Left Case
  if (balance < -1 && getBalance(root->right) > 0)
  {    root->right = rightRotate(root->right);
      return leftRotate(root);}
  return root;}
```

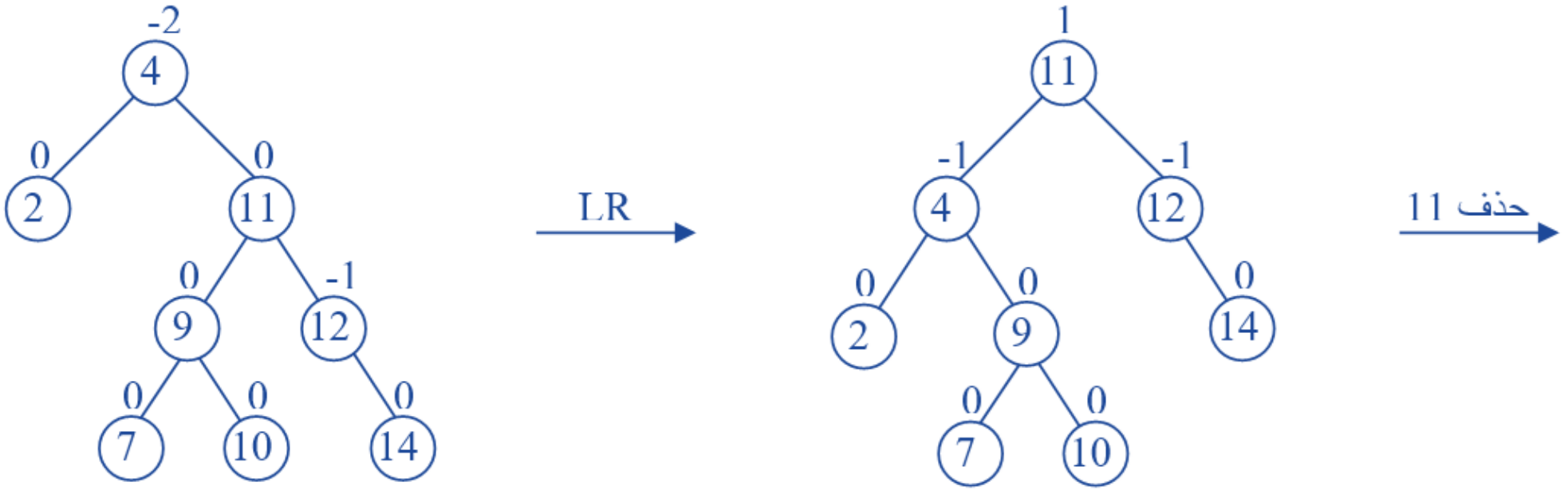
الحذف في أشجار AVL

- مثال:



الحذف في أشجار AVL

- مثال:



الحذف في أشجار AVL

- مثال: